CURSO DE ESPECIALIZAÇÃO EM AUTOMAÇÃO INDUSTRIAL

Fundamentos e Aplicações de Sistemas de Automação

Módulo 1:
Introdução aos Sistemas de Automação

Instrutor: Constantino Seixas Filho

18 de agosto de 2004

1

Livro

- Seixas, Constantino e Szuster, Marcelo, Programação Concorrente em ambiente WNT: Uma visão de automação, Editora da UFMG, 2003

Filosofia Geral – Modelo Hierárquico

Programação Concorrente

- Programação concorrente x Programação Seqüencial

Programação seqüencial [Gary Larson, The Far side Gallery]

Monotarefa X Multitarefas

UFMG

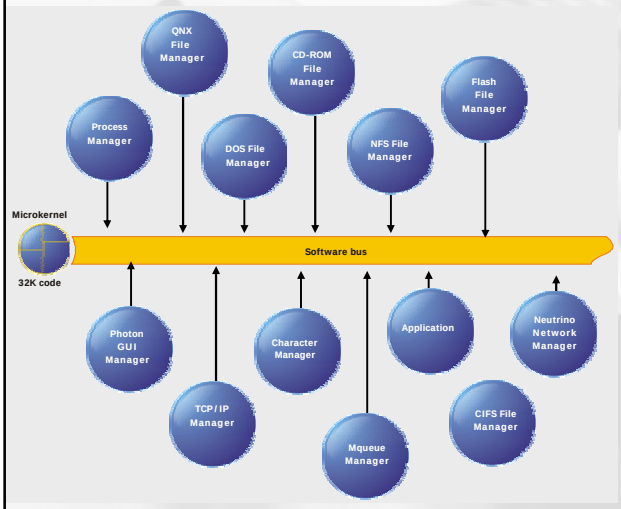
- Hoje tudo na vida é multitasking (multitarefas)
  - Trabalho, esporte, lazer, cuidar de filhos, etc.
- Só faz sentido quando não estamos utilizando o mesmo recurso da máquina

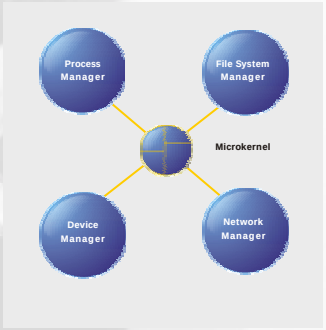
Por que programação concorrente ?

UFMG

- O computador é mais rápido que seus usuários
  Exemplo: digitação
  - Bom digitador: 60 palavras /minuto
  - Tempo de processamento de 1 caracter = 10 s
  - Computador realiza 10 s de trabalho a cada 2.3 dias
- Em geral várias atividades não concorrem pelos mesmos recursos: digitar, imprimir, ouvir mp3, baixar arquivos, compilar, etc.
- Melhor modelamento das atividades
- O próprio sistema operacional é formado por dezenas de processos concorrentes: arquitetura microkernel

Arquitetura Microkernel





9290-102-24-002 (Rev.:00)

Natureza das Aplicações

- Cíclicas (Evento de tempo)
- Em resposta a eventos de processo ou de operação
- Sincronizadas ao mudo externo
- De tempo real

Definições Básicas

- Escalonamento
- Scheduler = Escalonador
- Time-sharing
- Time-slice = Quantum
- Tasks: implementadas como processos ou threads
- Colaborativo x Preemptivo

Preempção

Chaveamento de Contexto

- A programação concorrente é mais ineficiente que a programação seqüencial. Por que ?
- Só faz sentido quando não estamos utilizando o mesmo recurso da máquina
- Processor bound (intensivo) x I/O bound

Instruções Atômicas

X : = X + 1;

CPU CISC	CPU RISC

inc <X>	ld AX, <X>
	inc <AX>
	ld <X>, AX

- Instruções atômicas são instruções que o processador executa como uma operação indivisível, isto é, sem interrupção

CISC x RISC

CISC = Complete Instruction Set Computer

- Instruction Set com grande repertório de instruções
- Instruções gastam vários clocks para completar
- Processador possui poucos registradores internos
- Instruções envolvem memória externa
- ALU para inteiros apenas
- Operações floating point executadas em co-processador externo
- Instruções utilizam microcódigo
- Exemplo: Intel 80386

CISC x RISC

RISC = Reduced Instruction Set Computer

- Instruction Set com pequeno repertório de instruções
- Instruções executam em apenas 1 ou menos ciclos de clock
- Processador possui muitos registradores internos
- Instruções buscam dados externos para registardor interno e operam com estes registardores
- Operações com inteiros e floating points
- Instruções não utilizam microcódigo
- Arquitetura pipeline e superescalar
- Exemplo: CPU Sparc SUN

Pipeline

- Várias operações simultâneas

9290-102-24-003 (Rev.:00)

- A CPU trabalha como uma linha de montagem

Superescalar

- Duplicando a linha de montagem

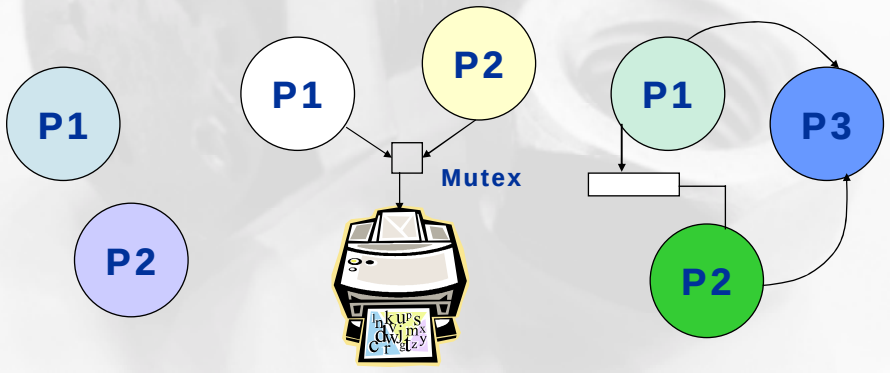
9290-102-24-003 (Rev.:00)

Mais Conceitos

- S.O. Multitasking ➔ Multiprogramação (multitasking)
- S.O. Multiusuário
- S.O. Multiprocessado ➔ Programação paralela
- Multiprocessamento Simétrico (SMP)
- Programação Distribuída

Interrelação entre Processos

- Não dependentes
- Contenção
- Cooperação



O Problema Fundamental da Programação Concorrente

Processo A	Processo B

int ConfirmaReserva(int PosicaoDesejada)
{ A1: if (MapaOcupacao[PosicaoDesejada]== LIVRE)
A2: MapaOcupacao[PosicaoDesejada]=OCUPADO;
else return(FALHA);
return(SUCESSO);
}
	int ConfirmaReserva(int PosicaoDesejada)
	{ B1: if (MapaOcupacao[PosicaoDesejada]== LIVRE)
	B2: MapaOcupacao[PosicaoDesejada]= OCUPADO;
	else return(FALHA);
	return(SUCESSO);
	}

L	L	L	O	L	L	L	L
O	L	L	O	O	L	O	O

9290-102-24 (rev.:00) 19

O Problema Fundamental da Programação Concorrente

Caso	Intercalação	Resultado
1	A1, A2, B1, B2	Programa A retorna SUCESSO e o Programa B retorna FALHA
2	B1, B2, A1, A2	Programa A retorna FALHA e o Programa B retorna SUCESSO
3	A1, B1, A2, B2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
4	B1, A1, A2, B2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
5	A1, B1, B2, A2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO
6	B1, A1, B2, A2	Programa A retorna SUCESSO e o Programa B retorna SUCESSO

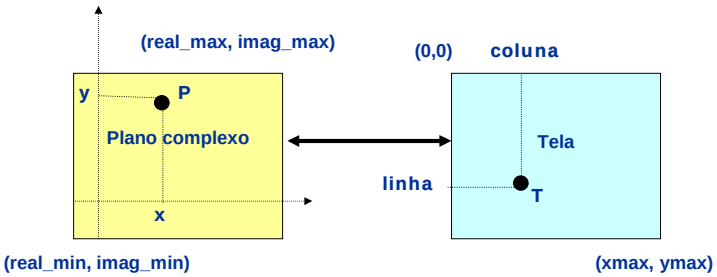
- Conclusão:
  Na programação concorrente não basta que cada programa seqüencial esteja logicamente correto. É necessário que produza um resultado correto para qualquer ordem de intercalação de suas instruções com uma outra instância do mesmo programa ou com qualquer outro programa em execução. Isto não exclui o próprio sistema operacional

9290-102-24 (rev.:00) 20

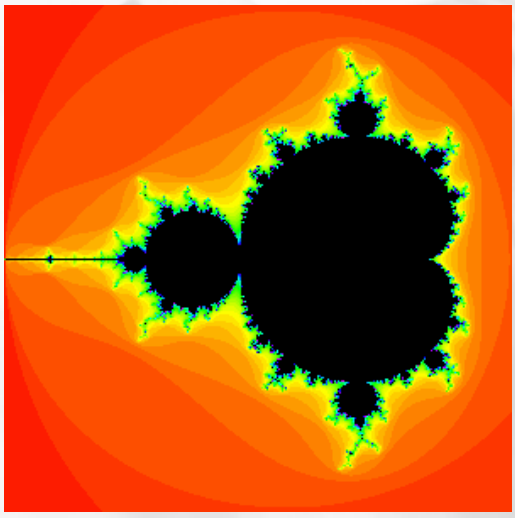
Exercícios: Mandelbrot Set

$$z_{n+1} = z^2{}_n + p$$

$$p = x + jy$$



Exercícios: Mandelbrot Set

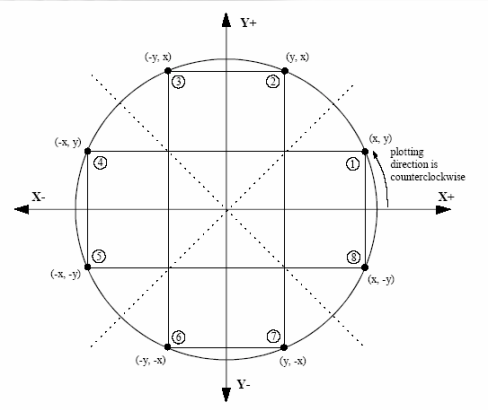


- Por que este problema é bom para ser dividido entre diversos processadores ?

Exercícios

- Como desenhar um círculo na tela do computador de forma eficiente e dividindo a tarefa entre pelo menos duas tarefas ?

$$X^2 + y^2 = r^2$$



Exercícios

```
procedure PlotCircle(CX, CY, R : longint);
begin
var X, Y : longint;
    XChange, YChange : longint;
    RadiusError : longint;
    X : R; !
    Y := 0;
    XChange : 1 2*R; ! "
    YChange : 1; !
    RadiusError := 0;
    while ( X Y ) do #
    begin
          Plot8CirclePoints(X,Y); {subroutine appears below}
          inc(Y);
          inc(RadiusError, YChange);
          inc(YChange,2);
          if ( 2*RadiusError + XChange > 0 ) then
          begin
              dec(X);
              inc(RadiusError, XChange);
              inc(XChange,2)
          end
    end
end; {procedure PlotCircle}
```

Exercícios

```
procedure Plot8CirclePoints(X,Y : longint);
begin
     PutPixel(CX+X, CY+Y); {point in octant 1}
     PutPixel(CX-X, CY+Y); {point in octant 4}
     PutPixel(CX-X, CY-Y); {point in octant 5}
     PutPixel(CX+X, CY-Y); {point in octant 8}
     PutPixel(CX+Y, CY+X); {point in octant 2}
     PutPixel(CX-Y, CY+X); {point in octant 3}
     PutPixel(CX-Y, CY-X); {point in octant 6}
     PutPixel(CX+Y, CY-X)  {point in octant 7}
end; {procedure Plot8CirclePoints}
```


Exercícios - Sort

- Colocar os números abaixo em ordem crescente usando um algoritmo que permita dividir a tarefa entre dois ou mais processos. Qual o algoritmo clássico mais aderente a este problema ?
- N = { 737, 4, 4354, 22, 54, 742, 672, 223, 42, 3, 77, 122, 4554, 675, 1, 564, 12, 343 }

Exercícios - Sort

- Colocar os números abaixo em ordem crescente usando um algoritmo que permita dividir a tarefa entre dois ou mais processos. Qual o algoritmo clássico mais aderente a este problema ?

- N = { 737, 4, 4354, 22, 54, 742, 672, 223, 42, 3, 77, 122, 4554, 675, 1, 564, 12, 343 }

Exercícios - Escalonamento

- Um casal deve realizar três tarefas: cortar a grama, passar o aspirador no chão e dar banho e trocar o bebê. Todas as três tarefas levam trinta minutos para ser realizadas e podem ser executadas por uma pessoa individualmente. Como conseguir a melhor alocação destas tarefas no tempo de tal forma a terminar o mais rápido possível ?

Muito Obrigado

Perguntas?

Constantino Seixas Filho

constantino.seixas@task.com.br